Почему распределенный SQL сложнее, чем кажется

Станислав Лукьянов

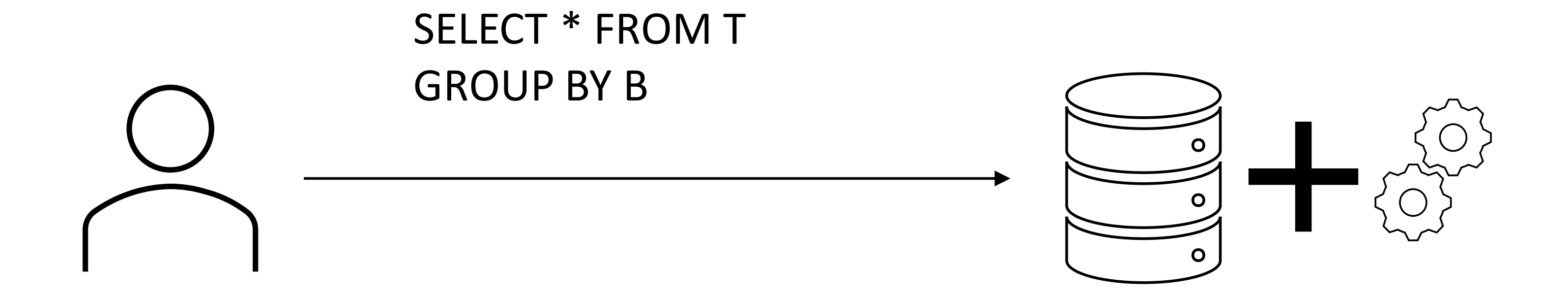




Распределенный SQL:
Что, Как и Зачем?



Распределенный SQL – Что?





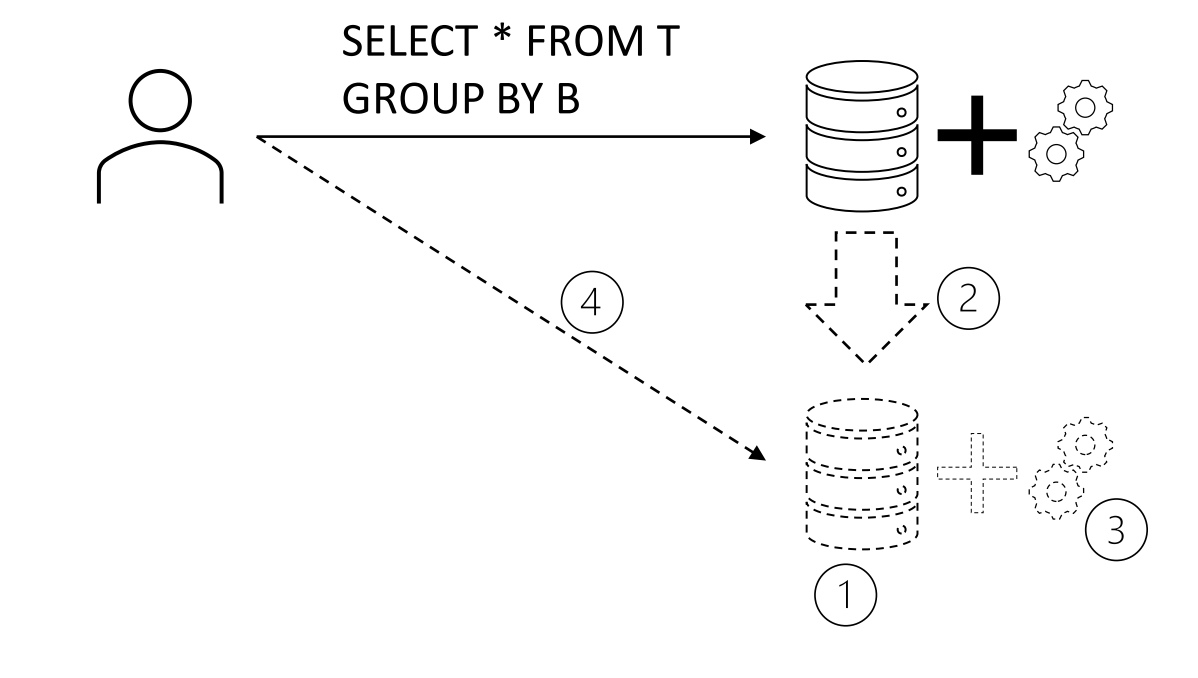
Распределенный SQL – Что?

SELECT * FROM T
GROUP BY B



Распределенный SQL – Как?

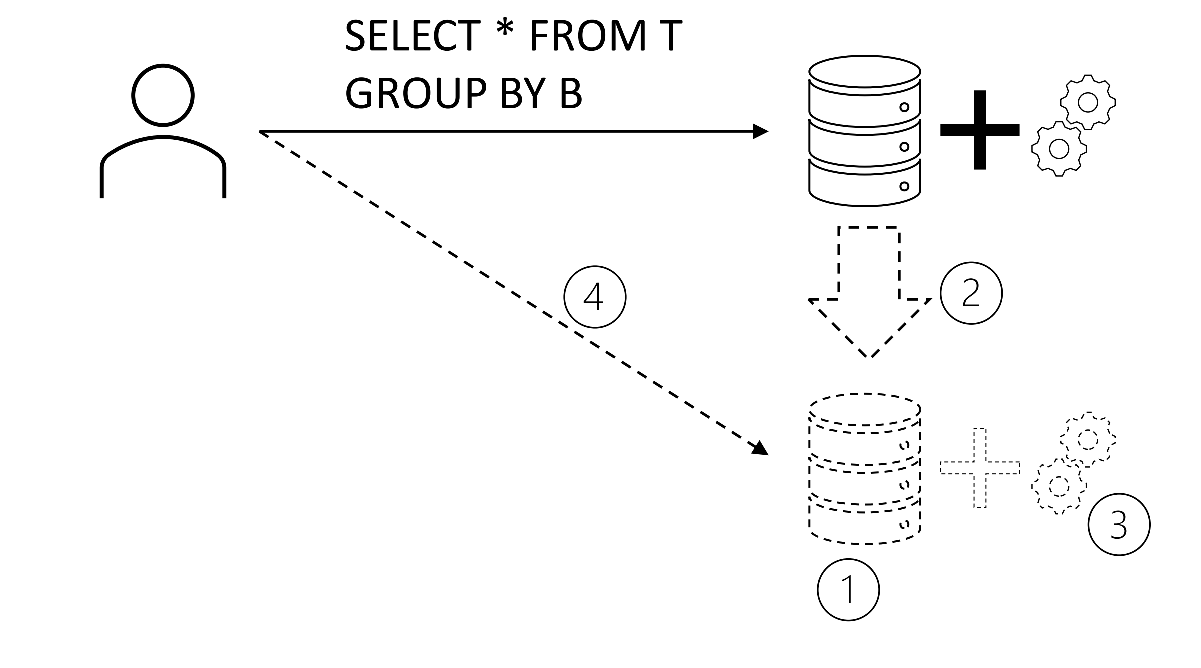
1. Как распределить данные?
   - Шардирование, репликация, колокация
2. Как реплицировать данные?
   - Active-passive vs active-active
   - Strong or eventual consistency
   - 2PC, PAXOS, MVCC, etc.
3. Как распределить вычисления?
   - Этапы вычислений, роли узлов
4. Как выполнить запрос?





Распределенный SQL – Как?

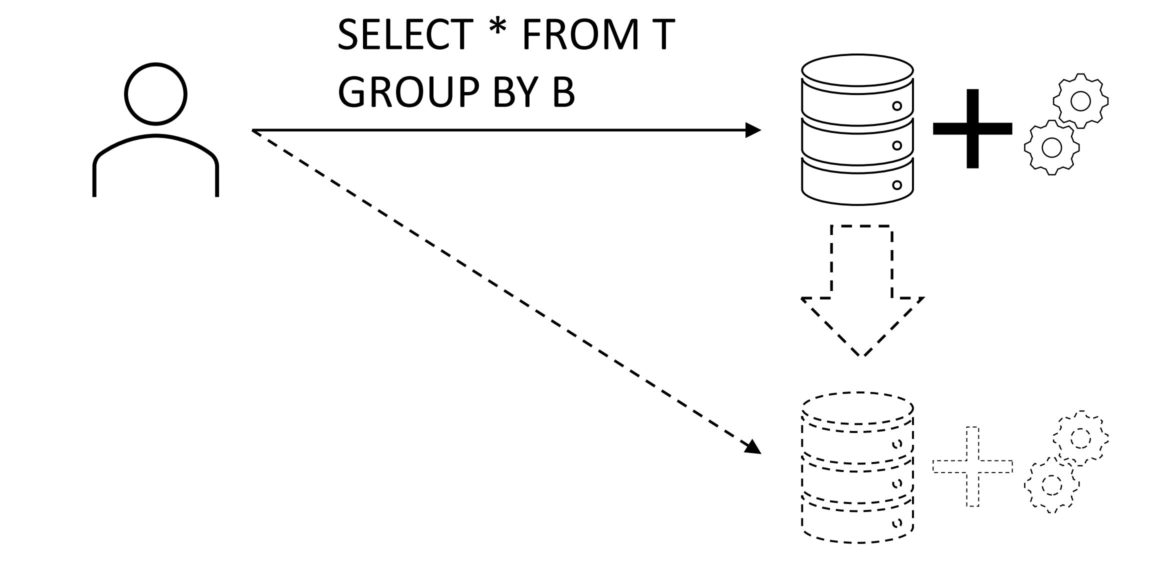
1. Как распределить данные?
   - Шардирование, репликация, колокация
2. Как реплицировать данные?
   - Active-passive vs active-active
   - Strong or eventual consistency
   - 2PC, PAXOS, MVCC, etc.
3. Как распределить вычисления?
   - Этапы вычислений, роли узлов
4. Как выполнить запрос?





Распределенный SQL – Зачем?

- Больше железа
  - Выше производительность
  - Больше емкость
- Несколько копий
  - Возможность распараллелить нагрузку
  - Ниже влияние потери отдельной машины
- Несколько регионов
  - Возможность локального доступа к данным
  - Ниже влияние отказа целого ЦОД-а



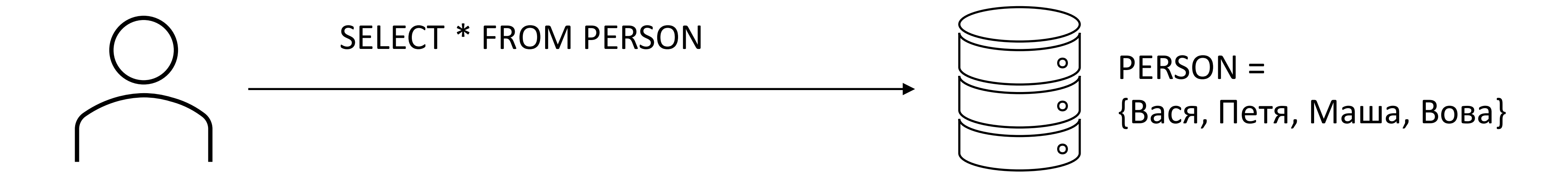


Распределение Данных

Как сохранить больше данных, чем влезает в RDBMS?



Репликация


SELECT * FROM PERSON
PERSON =
{Вася, Петя, Маша, Вова}


HL
HighLoad++
Весна 2021

Репликация

SELECT * FROM PERSON

PERSON =
{Вася, Петя, Маша, Вова}

PERSON =
{Вася, Петя, Маша, Вова}

PERSON =
{Вася, Петя, Маша, Вова}



Репликация

SELECT * FROM
PERSON

Load
Balancer

PERSON =
{Вася, Петя, Маша, Вова}

PERSON =
{Вася, Петя, Маша, Вова}

PERSON =
{Вася, Петя, Маша, Вова}



Репликация

SELECT * FROM PERSON

Load Balancer

PERSON = {Вася, Петя, Маша, Вова}

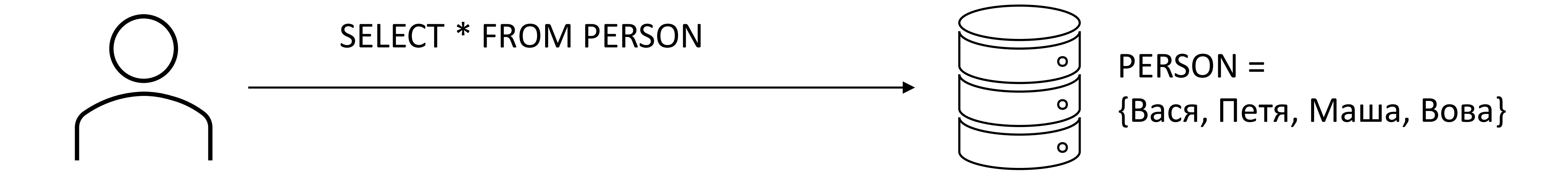
PERSON = {Вася, Петя, Маша, Вова}

PERSON = {Вася, Петя, Маша, Вова}

+ Доступность
– Масштабируемость

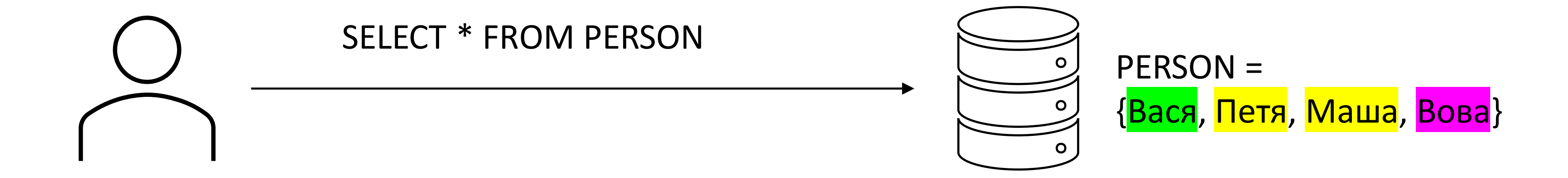


Шардирование





Шардирование





Шардирование

SELECT * FROM PERSON

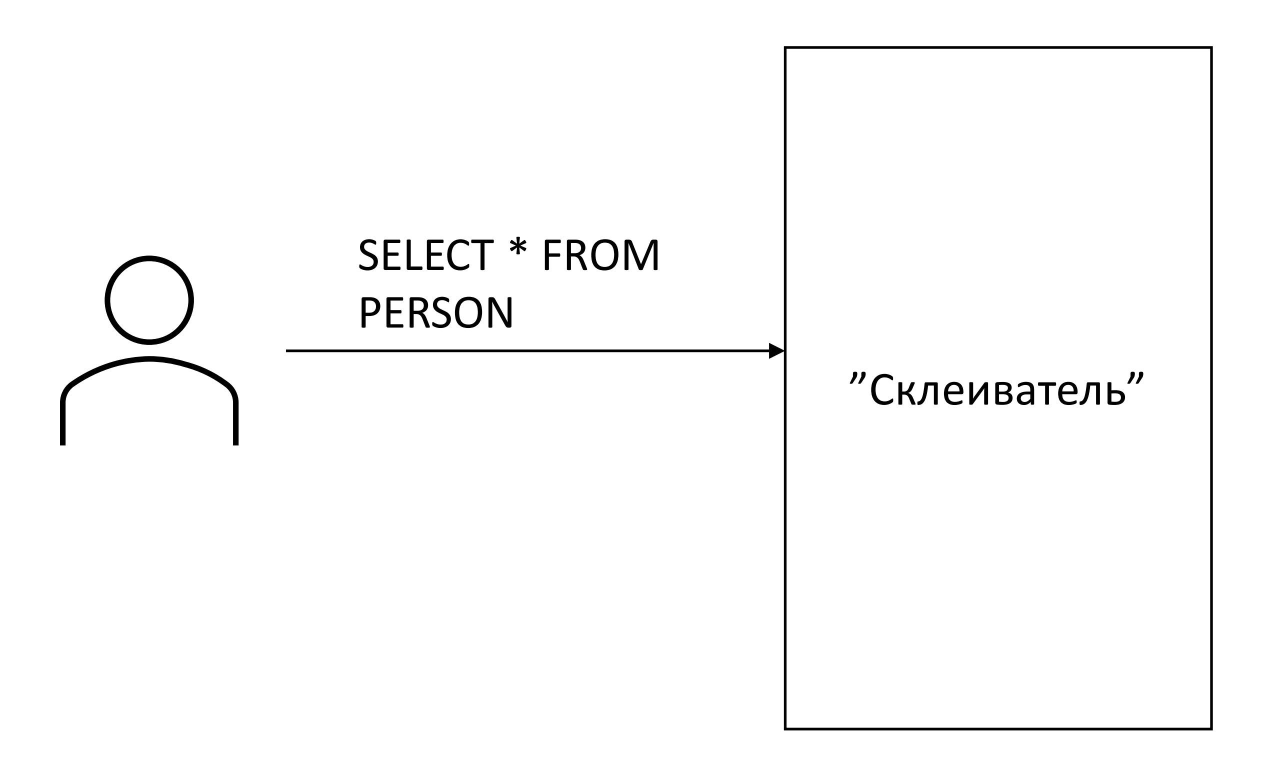
PERSON = {Вася}

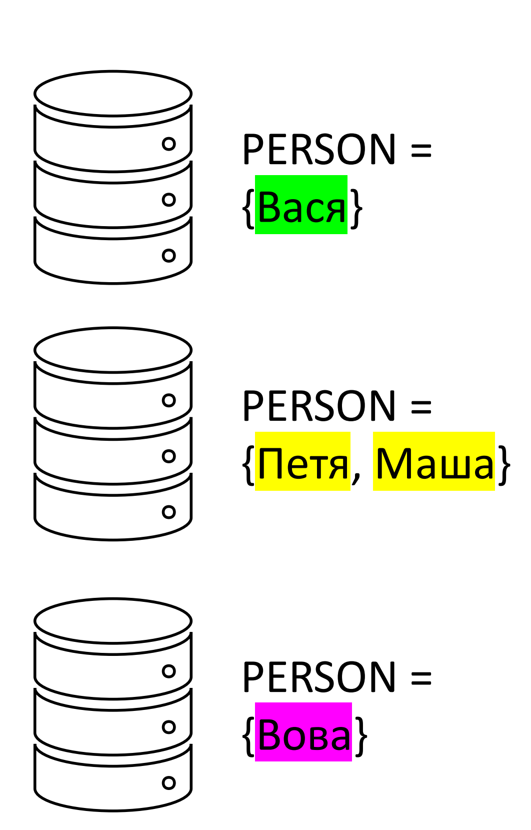
PERSON = {Петя, Маша}

PERSON = {Вова}



Шардирование







Шардирование

SELECT * FROM PERSON

"Склеиватель"

SELECT * FROM PERSON

PERSON = {Вася}

SELECT * FROM PERSON

PERSON = {Петя, Маша}

SELECT * FROM PERSON

PERSON = {Вова}



Шардирование

SELECT * FROM PERSON

"Склеиватель"

SELECT * FROM PERSON

{Вася}

PERSON = {Вася}

SELECT * FROM PERSON

{Петя, Маша}

PERSON = {Петя, Маша}

SELECT * FROM PERSON

{Вова}

PERSON = {Вова}



Шардирование

SELECT * FROM PERSON

{Вася, Петя, Маша, Вова}

"Склеиватель"

SELECT * FROM PERSON

{Вася}

PERSON = {Вася}

SELECT * FROM PERSON

{Петя, Маша}

PERSON = {Петя, Маша}

SELECT * FROM PERSON

{Вова}

PERSON = {Вова}



Шардирование

SELECT * FROM PERSON

{Вася, Вова}

"Склеиватель"

SELECT * FROM PERSON

{Вася}

PERSON = {Вася}

SELECT * FROM PERSON

{Петя, Маша}

PERSON = {Петя, Маша}

SELECT * FROM PERSON

{Вова}

PERSON = {Вова}



Шардирование

SELECT * FROM PERSON

{Вася, Вова}

"Склеиватель"

SELECT * FROM PERSON

{Вася}

PERSON = {Вася}

SELECT * FROM PERSON

{Петя …

…ERSON = {Петя, Маша}

SELECT * FROM PERSON

{Вова}

PERSON = {Вова}

– Доступность
+ Масштабируемость



Шардирование + Репликация

SELECT * FROM PERSON

{Вася, Петя, Маша, Вова}

”Склеиватель”

SELECT * FROM PERSON

{Вася}

PERSON = {Вася}

SELECT * FROM PERSON

{Петя, Маша}

PERSON = {Петя, Маша}

SELECT * FROM PERSON

{Вова}

PERSON = {Вова}



Шардирование + Репликация

SELECT * FROM PERSON

{Вася, Петя, Маша, Вова}

”Склеиватель”

SELECT * FROM PERSON

{Вася}

PERSON = {Вася, Петя, Маша}

SELECT * FROM PERSON

{Петя, Маша}

PERSON = {Петя, Маша, Вова}

SELECT * FROM PERSON

{Вова}

PERSON = {Вова, Вася}



Шардирование + Репликация

SELECT * FROM PERSON

{Вася, Петя, Маша, Вова}

"Склеиватель"

SELECT * FROM PERSON

{Вася, Петя, Маша}

PERSON = {Вася, Петя, Маша}

SELECT * FROM PERSON

{Петя, Маша}

PERSON = {Петя, Маша, Вова}

SELECT * FROM PERSON

{Вова}

PERSON = {Вова, Вася}



Шардирование + Репликация

SELECT * FROM PERSON

{Вася, Петя, Маша, Вова}

"Склеиватель"

SELECT * FROM PERSON

{Вася, Петя, Маша}

PERSON = {Вася, Петя, Маша}

SELECT * FROM PERSON

PERSON = {Петя, Маша, Вова}

SELECT * FROM PERSON

{Вова}

PERSON = {Вова, Вася}

+ Доступность
+ Масштабируемость



Распределение Данных в Apache Ignite

- Все таблицы шардированы
  - Шарды называются **партициями**
  - Число партиций настраивается (**1024** по умолчанию)
  - Номер партиции – **хэш** от ключа
- Два режима таблиц
  - PARTITIONED – заданное количество backup-копий

```
CREATE TABLE PERSON (ID INT PRIMARY KEY, NAME VARCHAR)
WITH “template=partitioned, backups=1”
```

  - RERPLICATED – количество копий партиций всегда равно количеству узлов

```
CREATE TABLE PERSON (ID INT PRIMARY KEY, NAME VARCHAR)
WITH “template=replicated”
```

- Распределение партиций по узлам – автоматическое
  - Используется **rendezvous hashing**
  - Партиции автоматически **переезжают** при входе-выходе узлов
  - У каждой партиции один **primary**-узел и 0..N-1 **backup**-узлов



Выполнение Запросов

Что Делает «Склеиватель»?



Выполнение Запросов

”Склеиватель”



Map-Reduce

Reduce

Map

”Склеиватель”



Map-Reduce

Reduce

Map

"Склеиватель"

- Scan
- Фильтрация



Map-Reduce

Reduce

Map

”Склеиватель”

- JOIN
- Агрегация

- Scan
- Фильтрация



Map-Reduce

Reduce

Map

"Склеиватель"

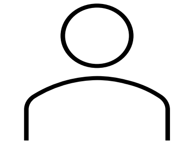
- Финальная склейка

- JOIN
- Агрегация

- Scan
- Фильтрация



Map-Reduce в Apache Ignite



Reduce

Reducer

Map

Mappers



Map-Reduce в Apache Ignite

Reduce

Map

SELECT *
FROM PERSON

Mappers

Reducer



Map-Reduce в Apache Ignite

Reduce

Map

SELECT *
FROM PERSON

Query
Splitter

Mappers

Reducer



Map-Reduce в Apache Ignite

Reduce

Map

SELECT *
FROM PERSON

Query Splitter

SELECT *
FROM PERSON

SELECT *
FROM T1

Mappers

Reducer



Map-Reduce в Apache Ignite

Reduce

Map

Query
Splitter

SELECT *
FROM PERSON

SELECT *
FROM PERSON

SELECT *
FROM T1

Mappers

Reducer



Map-Reduce в Apache Ignite

Reduce

Map

SELECT *
FROM PERSON

Query Splitter

SELECT *
FROM PERSON

SELECT *
FROM T1

Mappers

T1 (Virtual)

Reducer



Map-Reduce в Apache Ignite

Reduce

Map

SELECT *
FROM PERSON

Query
Splitter

SELECT *
FROM PERSON

SELECT *
FROM T1

T1 (Virtual)

Reducer

Mappers



Map-Reduce в Apache Ignite

Reduce

Map

SELECT *
FROM PERSON

Query
Splitter

SELECT *
FROM PERSON

SELECT *
FROM T1

T1 (Virtual)

Reducer

Mappers



Map-Reduce в Apache Ignite

- Reducer контролирует выполнение запроса
  - Парсит и «разбивает» запрос на map- и reduce-фазы
  - Отправляет map-запросы
  - Создает виртуальные таблицы для результатов map
- Reducer’ом могут выступать разные узлы
  - Один из серверов с данными
  - Выделенный узел
  - Приложение
- Mapper работает как обычная RDBMS
  - Парсинг, индексы, оптимизации, etc.



Преобразование SQL для Map-Reduce

Как Минимизировать Фазу Reduce?



Наивный Map-Reduce

Reduce

Map

Query
Splitter

SELECT *
FROM PERSON

SELECT *
FROM T1

SELECT *
FROM PERSON

Mappers

T1 (Virtual)

Reducer



Наивный Map-Reduce

Reduce

Map

Query
Splitter

SELECT *
FROM PERSON

SELECT **COUNT(*)**
FROM PERSON

SELECT **COUNT(*)**
FROM T1

Mappers

T1 (Virtual)

Reducer



Наивный Map-Reduce

Reduce

Map

SELECT **COUNT(*)**
FROM PERSON

Query
Splitter

SELECT *
FROM PERSON

SELECT **COUNT(*)**
FROM T1

T1 (Virtual)

Mappers

Reducer

Плохо – пересылаем много данных!



Считаем Агрегаты на Map

Reduce

Map

Query Splitter

SELECT **COUNT(*)**
FROM PERSON

SELECT **COUNT(*)**
FROM PERSON

SELECT **SUM(1)**
FROM T1

Mappers

T1 (Virtual)

Reducer



Считаем Агрегаты на Map

Reduce

Map

Query Splitter

SELECT **COUNT(*)**
FROM PERSON

SELECT **SUM(1)**
FROM T1

SELECT **COUNT(*)**
FROM PERSON

Mappers

T1 (Virtual)

Reducer

Гораздо лучше!



Разбиение Простых Запросов

Исходный Запрос	**Map**	**Reduce**
SELECT * FROM PERSON **WHERE** NAME = “Вася”	SELECT * FROM PERSON WHERE NAME = “Вася”	SELECT * FROM T1
SELECT **COUNT(*)** FROM PERSON	SELECT COUNT(*) FROM PERSON	SELECT SUM(1) FROM T1
SELECT **AVG(AGE)** FROM PERSON	SELECT SUM(AGE) as SUM_AGE, COUNT(AGE) as CNT FROM PERSON	SELECT SUM(SUM_AGE) / SUM(CNT) FROM T1

- WHERE обычно можно сделать полностью на map
- Агрегации можно частично посчитать на map и досчитать на reduce
    - Каждый вид агрегации требует своей логики



Распределенный JOIN

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```

PERSON	
NAME	COMP_ID
Вася	1
Петя	2
Маша	3
Вова	3

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



Распределенный JOIN

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```

Ожидаемый результат:

Вася, APPL
Петя, GOOGL
Маша, FB
Вова, FB

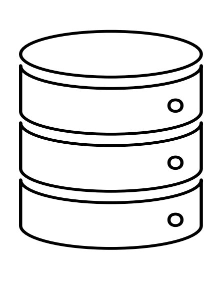
PERSON	
NAME	COMP_ID
Вася	1
Петя	2
Маша	3
Вова	3

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



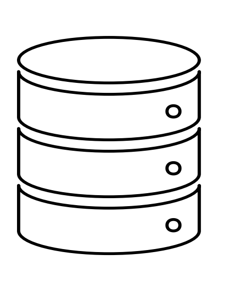
Распределенный JOIN

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```



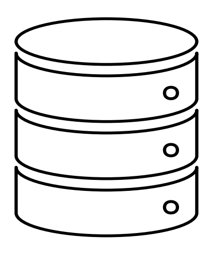
PERSON	
NAME	COMP_ID
Вася	1

COMPANY	
NAME	ID
APPL	1



PERSON	
NAME	COMP_ID
Петя	2
Маша	3

COMPANY	
NAME	ID
GOOGL	2



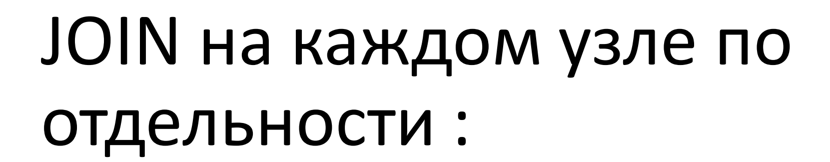
PERSON	
NAME	COMP_ID
Вова	3

COMPANY	
NAME	ID
FB	3

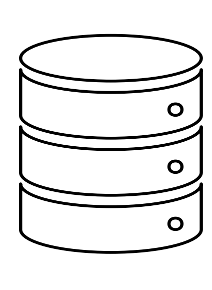


Распределенный JOIN

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```

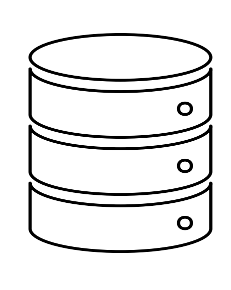

JOIN на каждом узле по отдельности :

Вася, APPL
Петя, GOOGL
Вова, FB



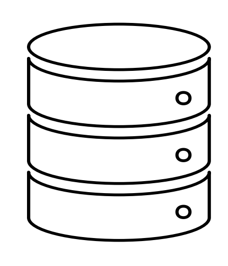
PERSON	
NAME	COMP_ID
Вася	1

COMPANY	
NAME	ID
APPL	1



PERSON	
NAME	COMP_ID
Петя	2
Маша	3

COMPANY	
NAME	ID
GOOGL	2



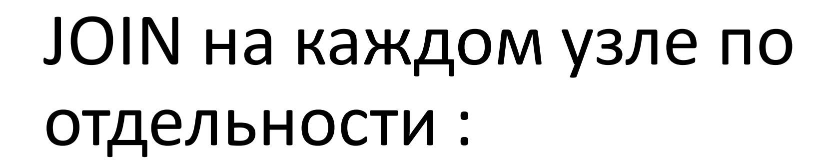
PERSON	
NAME	COMP_ID
Вова	3

COMPANY	
NAME	ID
FB	3

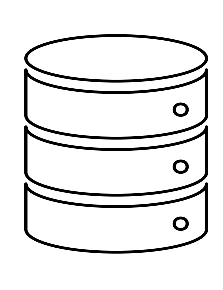


Распределенный JOIN

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```

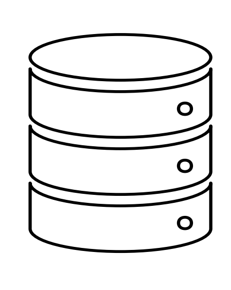


Вася, APPL
Петя, GOOGL
Вова, FB **Ошибка!**



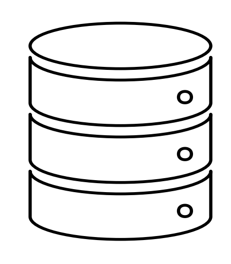
PERSON	
NAME	COMP_ID
Вася	1

COMPANY	
NAME	ID
APPL	1



PERSON	
NAME	COMP_ID
Петя	2
Маша	3

COMPANY	
NAME	ID
GOOGL	2



PERSON	
NAME	COMP_ID
Вова	3

COMPANY	
NAME	ID
FB	3



Распределенный JOIN

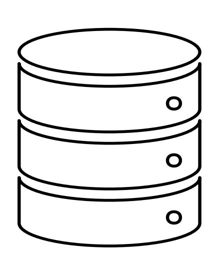
- Делать на reduce – плохо масштабируется
- Делать на map – не работает

- Можно ли заставить JOIN работать на map?
- Да – с помощью колокации



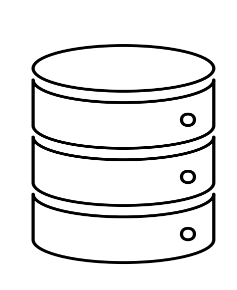
Колокация Данных по Ключу

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```



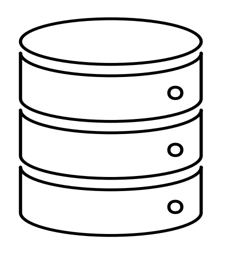
PERSON	
NAME	COMP_ID
Вася	1

COMPANY	
NAME	ID
APPL	1



PERSON	
NAME	COMP_ID
Петя	2

COMPANY	
NAME	ID
GOOGL	2



PERSON	
NAME	COMP_ID
Вова	3
Маша	3

COMPANY	
NAME	ID
FB	3

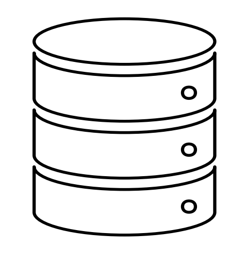


Колокация Данных по Ключу

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```

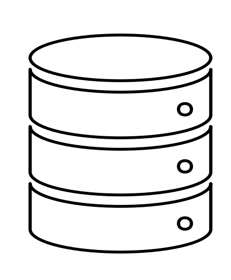
Колокация по COMPANY.ID и PERSON.COMP_ID:





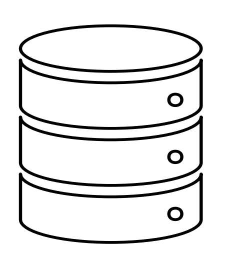
PERSON	
NAME	COMP_ID
Вася	1

COMPANY	
NAME	ID
APPL	1



PERSON	
NAME	COMP_ID
Петя	2

COMPANY	
NAME	ID
GOOGL	2



PERSON	
NAME	COMP_ID
Вова	3
Маша	3

COMPANY	
NAME	ID
FB	3



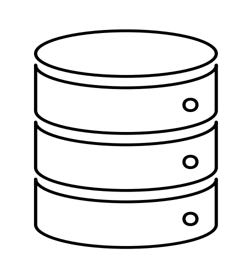
Колокация Данных по Ключу

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```

Колокация по COMPANY.ID и PERSON.COMP_ID:

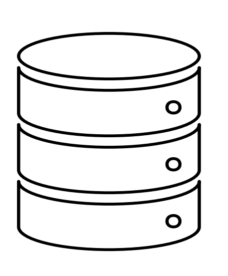






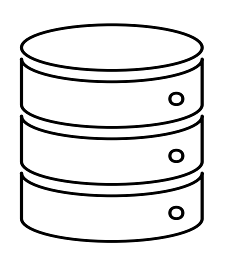
PERSON	
NAME	COMP_ID
Вася	1

COMPANY	
NAME	ID
APPL	1



PERSON	
NAME	COMP_ID
Петя	2

COMPANY	
NAME	ID
GOOGL	2



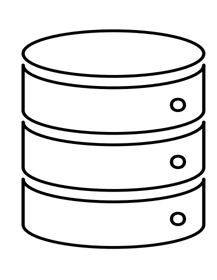
PERSON	
NAME	COMP_ID
Вова	3
Маша	3

COMPANY	
NAME	ID
FB	3



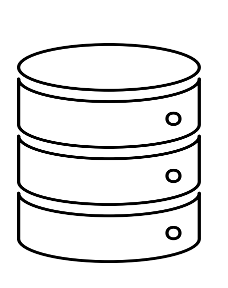
Колокация с Помощью Репликации

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```



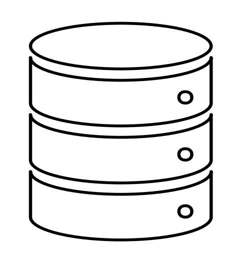
PERSON	
NAME	COMP_ID
Вася	1

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



PERSON	
NAME	COMP_ID
Петя	2

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



PERSON	
NAME	COMP_ID
Вова	3
Маша	3

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



Колокация с Помощью Репликации

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```

Шардированный PERSON +
реплицированный COMPANY:

Вася, APPL
Петя, GOOGL
Маша, FB
Вова, FB

PERSON	
NAME	COMP_ID
Вася	1

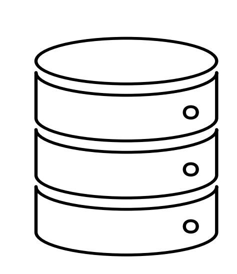
COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3





PERSON	
NAME	COMP_ID
Петя	2

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



PERSON	
NAME	COMP_ID
Вова	3
Маша	3

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



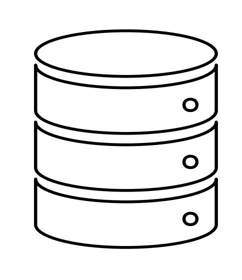
Колокация с Помощью Репликации

```
SELECT P.NAME, C.NAME
FROM PERSON P
JOIN COMPANY C
ON P.COMP_ID = C.ID
```

Шардированный PERSON + реплицированный COMPANY:

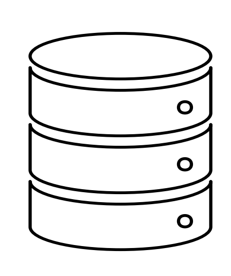
Вася, APPL
Петя, GOOGL
Маша, FB
Вова, FB





PERSON	
NAME	COMP_ID
Вася	1

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



PERSON	
NAME	COMP_ID
Петя	2

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



PERSON	
NAME	COMP_ID
Вова	3
Маша	3

COMPANY	
NAME	ID
APPL	1
GOOGL	2
FB	3



Колокация Данных

- По ключу
  - Подходит, когда все данные относятся к одной центральной сущности
    - Аккаунт, транзакция, филиал
- С помощью репликации
  - Подходит для таблиц-справочников
    - Схема «звезда»
- Колокация помогает ускорить распределенные JOIN, GROUP BY, подзапросы



Оптимизация Reduce

1. Наивно пытаемся спихнуть работу на map
   - Подходит для WHERE по полям таблицы
2. Хитро разбиваем агрегацию на map-reduce
   - COUNT, AVG, MIN, MAX, etc. несложно разбить
   - DISTINCT и ORDER BY сложнее
3. Колоцируем данные
   - Помогает с JOIN и агрегацией
   - Требует подготовки схемы БД
4. Оставшееся делаем на reduce
   - Если осталось много, то работать будет медленно

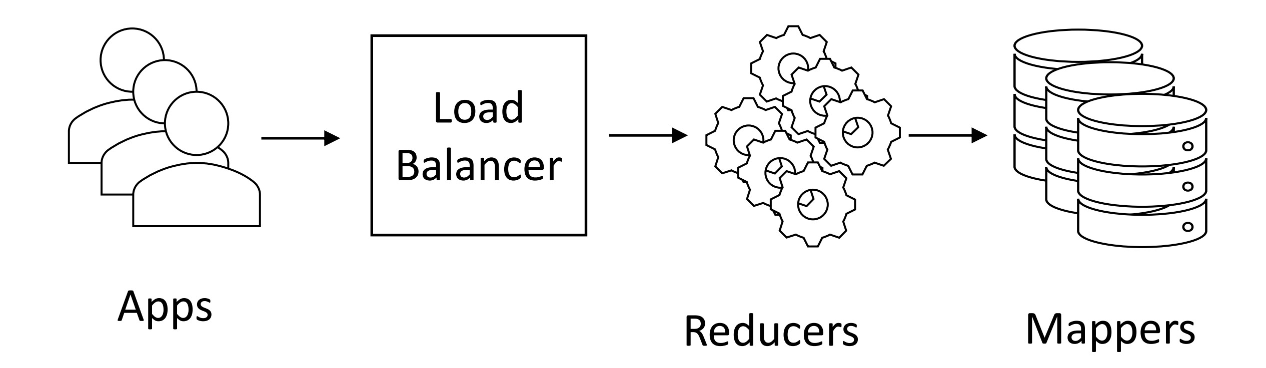


Если Reduce Сложный



Если Reduce Сложный

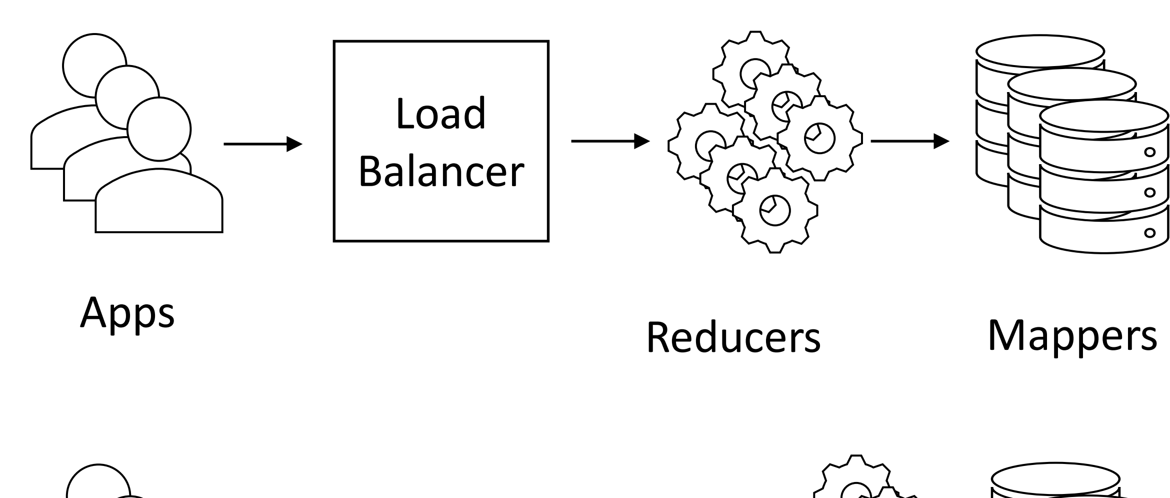
- Решение 1: несколько reducer’ов
  - Работает, если есть, что балансировать





Если Reduce Сложный

- Решение 1: несколько reducer'ов
  - Работает, если есть, что балансировать



- Решение 2: многофазные запросы





Оптимизация Reduce в Apache Ignite

- Базовые оптимизации функций агрегации
- JOINs считаются сколоцированными по умолчанию
  - Неправильная модель колокации – неправильный результат
  - Опции затянуть все данные на reduce нет даже в хинтах
  - Трехфазные запросы для несколоцированных JOINs
- В разработке: новый движок SQL на Apache Calcite
  - Apache Ignite 3.0-alpha2 в мае 2021



Колокация в Apache Ignite

- По ключу

```
CREATE TABLE PERSON (ID INT PRIMARY KEY, NAME VARCHAR,
COMP_ID INT)
WITH "template=partitioned, affinity_key=COMP_ID"
CREATE TABLE COMPANY (ID INT PRIMARY KEY, NAME VARCHAR)
WITH "template=partitioned"
```

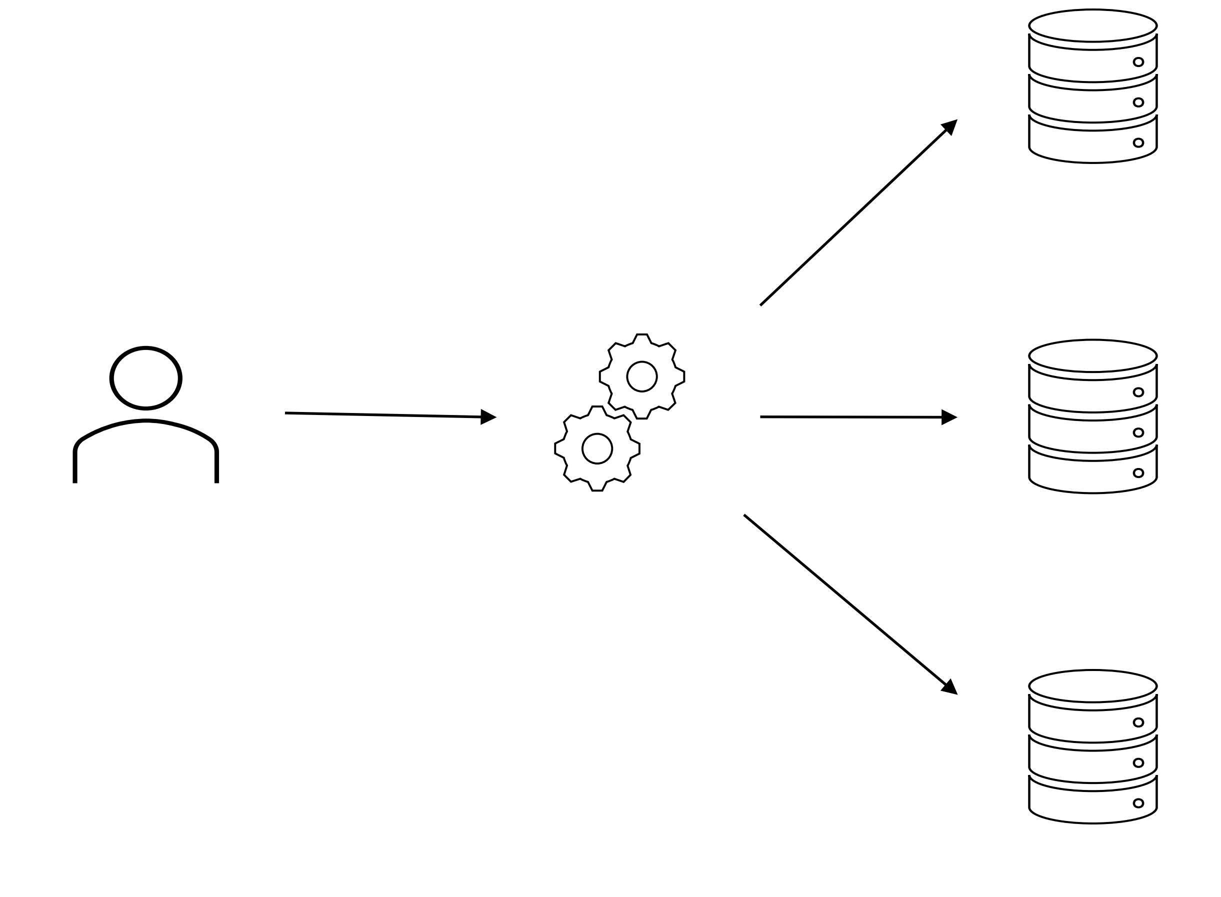
- С помощью репликации

```
CREATE TABLE PERSON (ID INT PRIMARY KEY, NAME VARCHAR,
COMP_ID INT)
WITH "template=partitioned"
CREATE TABLE COMPANY (ID INT PRIMARY KEY, NAME VARCHAR)
WITH "template=replicated"
```




Несколоцированные JOINs в Apache Ignite

«Расширенный» map:

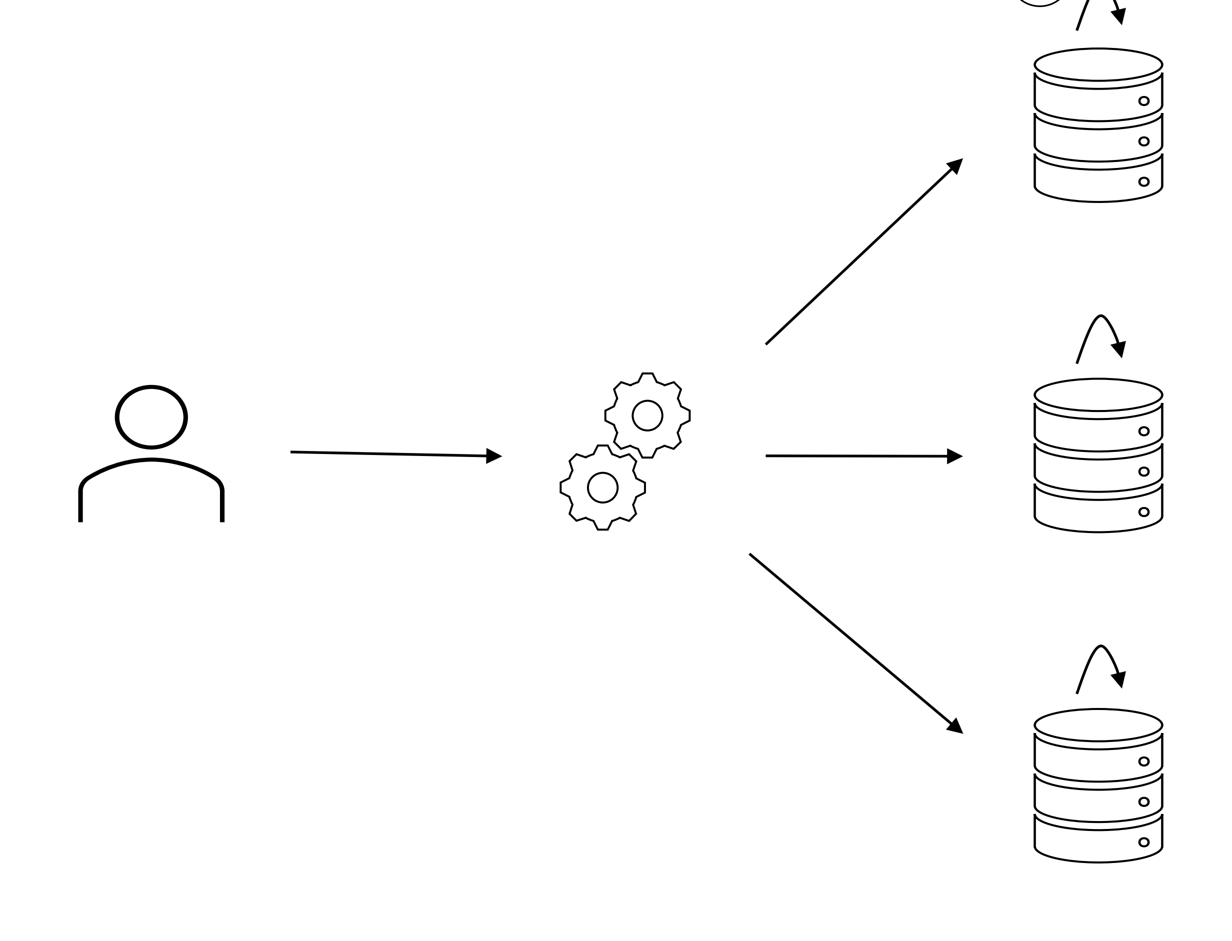




Несколоцированные JOINs в Apache Ignite

«Расширенный» map:

1. Mappers сканируют левую таблицу локально

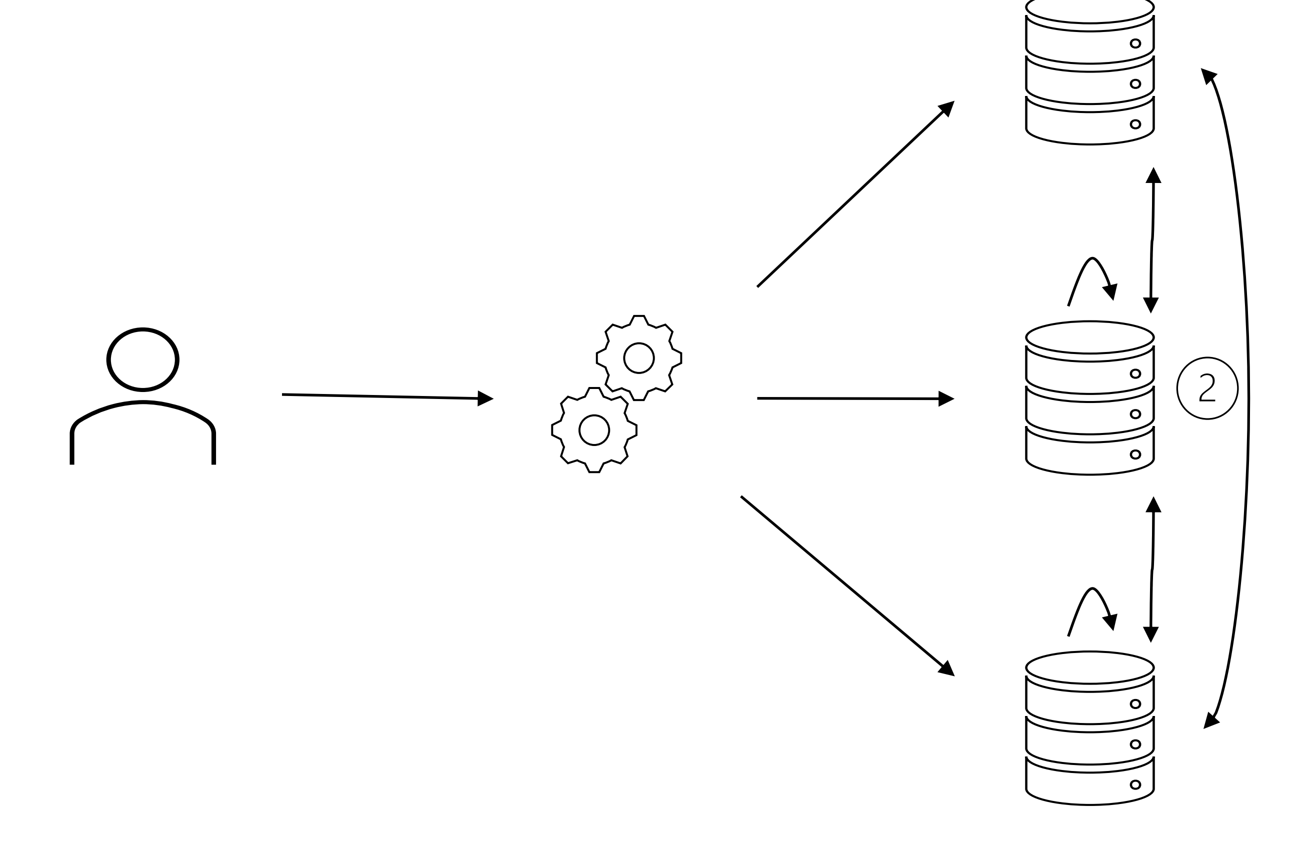




Несколоцированные JOINs в Apache Ignite

«Расширенный» map:

1. Mappers сканируют левую таблицу локально
2. Mappers запрашивают данные правой таблицы у других mappers

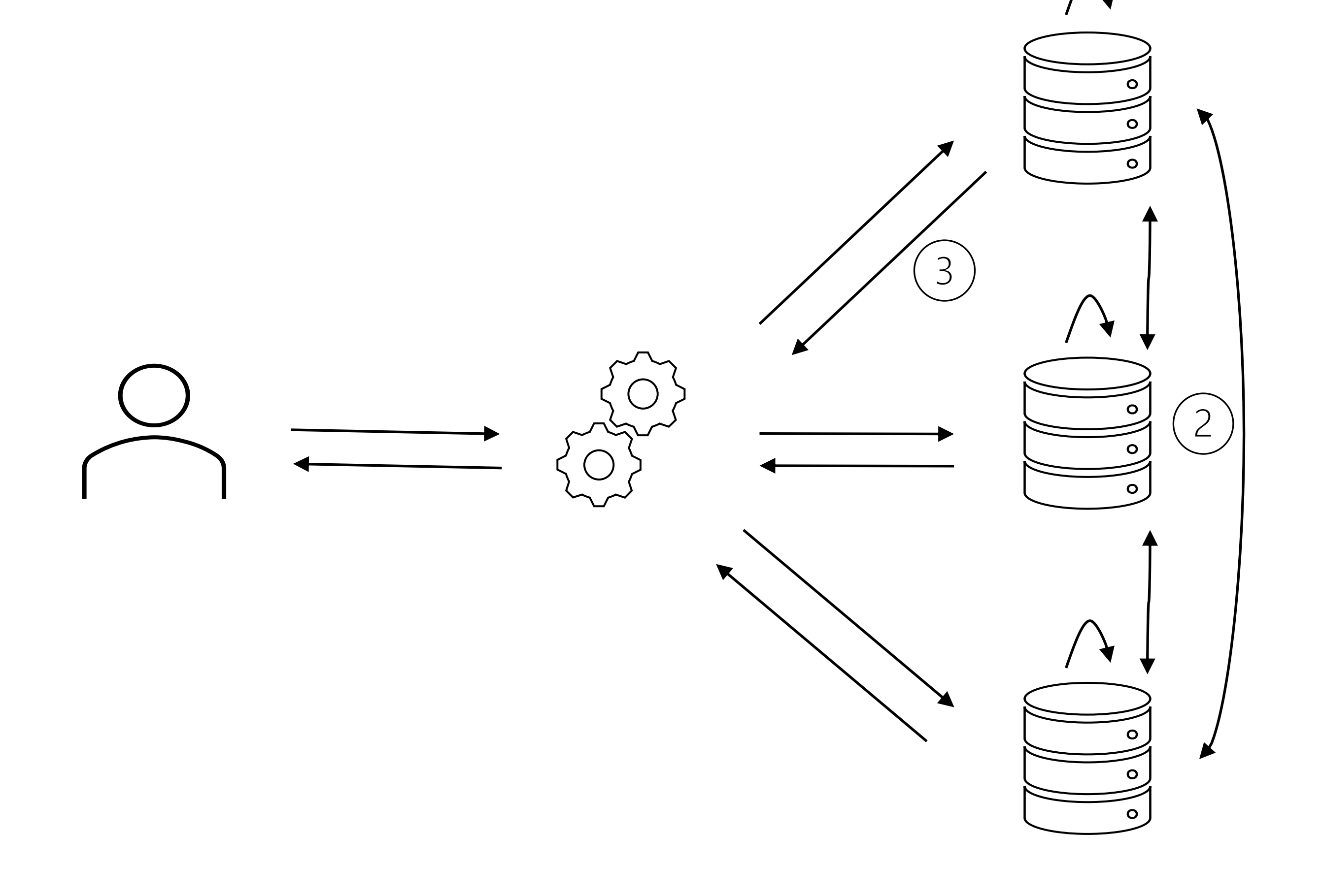




Несколоцированные JOINs в Apache Ignite

«Расширенный» map:

1. Mappers сканируют левую таблицу локально
2. Mappers запрашивают данные правой таблицы у других mappers
3. Mappers делают JOIN и возвращают результат





О Чем Поговорили

- Для масштабирования нужно шардирование…
- …которое требует склейки частичных результатов…
- …которую можно делать по map-reduce…
- …но reduce – это bottleneck…
- …с которым можно бороться:
  - Фильтрация на map
  - Хитрые разбиения агрегации
  - Колокация данных по ключу и с помощью репликации
  - Многофазные вычисления



Выводы

- Нужно понимать, как ваша распределенная база выполняет сложные запросы
- JOIN и агрегации могут плохо масштабироваться даже в распределенной системе
- Колокация – круто, но нужно быть готовым к тюнингу модели данных и запросов



Q&A

Спроси меня:

- stanlukyanov@gmail.com
- Telegram: @lukyanovsa

Apache Ignite Community:

- ignite.apache.org
- dev@ignite.apache.org

